DW_ram_r_w_2c_dff

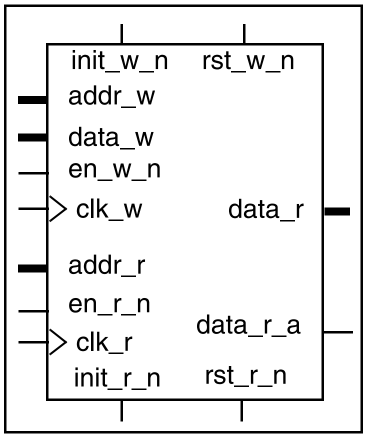
Synchronous Two-Clock RAM (Flip-Flop-Based)

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Dual port RAM with independent write clock domain and read clock domain ports
- Parameterized word depth
- Parameterized data width
- Parameterized memory access modes
- Synchronous static memory
- Separate synchronous and asynchronous reset control



Description

The DW_ram_r_w_2c_dff component is a parameterized synchronous two-clock domain dual-port static RAM. The write port is synchronous to the write domain clock and the read port is synchronous to the read domain clock.

The RAM is capable of having its write interface retimed prior to storing the incoming data content as well as having its read interface retimed prior to accessing the stored contents. Also, the data read out can be retimed before leaving the component. These capabilities are under parameter control.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk_w	1 bit	Input	Write domain clock
rst_w_n	1 bit	Input	Asynchronous reset in write clock domain, active low
init_w_n	1-bit	Input	Synchronous reset in write clock domain, active low
en_w_n	1 bit	Input	Write enable, active low
addr_w	*addr_width* bits	Input	Write address bus
data_w	*width* bits	Input	Write data in
clk_r	1 bit	Input	Read domain clock
rst_r_n	1 bit	Input	Asynchronous reset in read clock domain, active low
init_r_n	1-bit	Input	Synchronous reset in read clock domain, active low
en_r_n	1 bit	Input	Read enable, active low

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
addr_r	*addr_width* bits	Input	Read address bus
data_r_a	1 bit	Output	Read data arrival status, active high
data_r	*width* bits	Output	Read data out

Table 1-2 Parameter Description

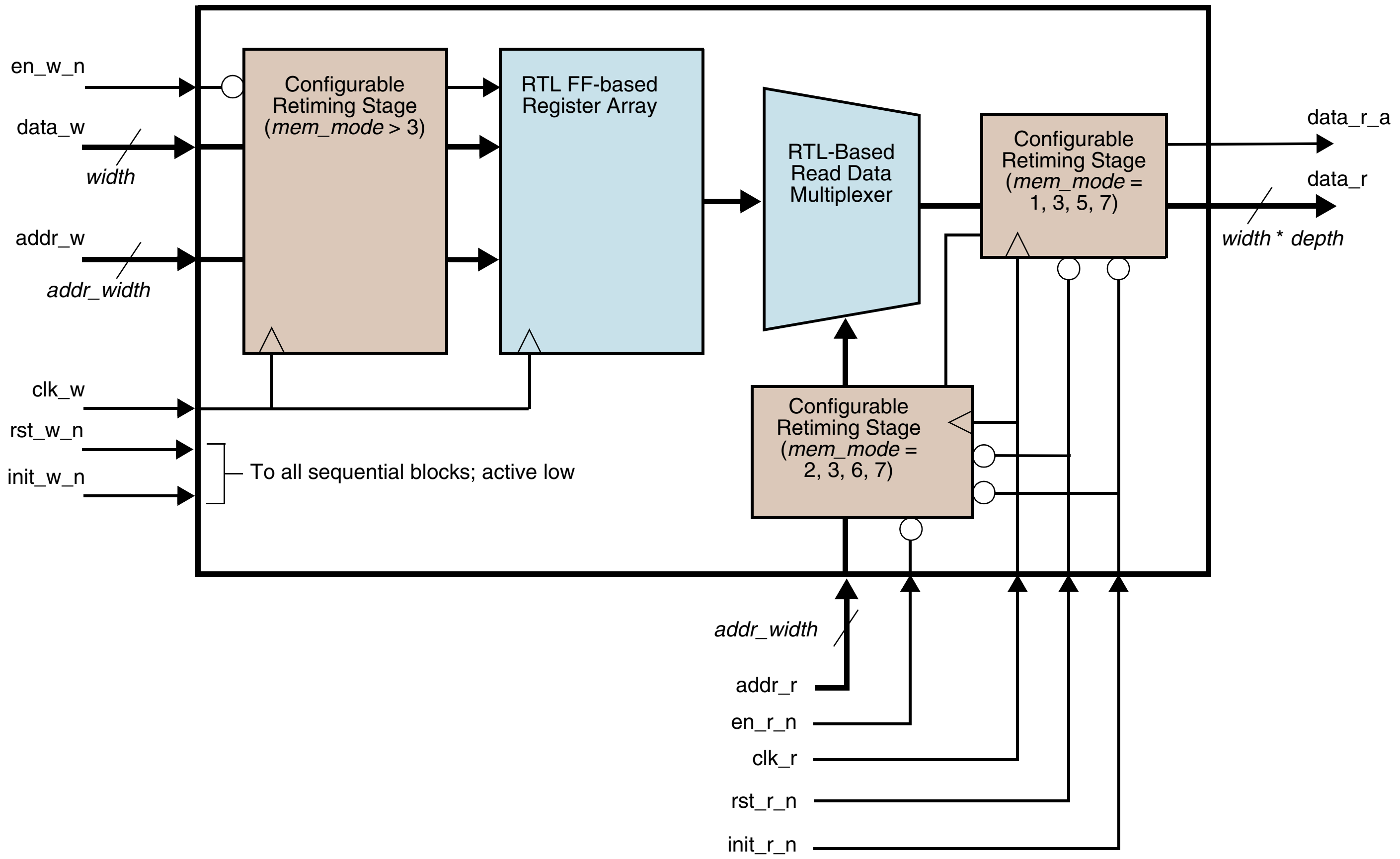
Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input `data_w` and output `data_r`
depth	2 to 1024 Default: 4	Desired number of RAM locations
addr_width	1 to 10 Default: 2	RAM address width ceil($\log_2$(*depth*))
mem_mode	0 to 7 Default: 1	Memory access control/datapath pipelining Defines where and how many retiming stages in RAM: ■ 0 = No pre- or post-retiming ■ 1 = RAM data out retiming (post-retiming) ■ 2 = RAM read address retiming (pre-retiming) ■ 3 = RAM data out and read address retiming ■ 4 = RAM write interface retiming (pre-retiming) ■ 5 = RAM write interface and RAM data out retiming ■ 6 = RAM write interface and read address retiming ■ 7 = RAM data out, write interface and read address retiming For details, see Figure 1-2 on page 4.
rst_mode	0 or 1 Default: 0	Reset mode ■ 0 = Resets clear RAM array ■ 1 = Resets do not clear RAM array

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Block Diagram

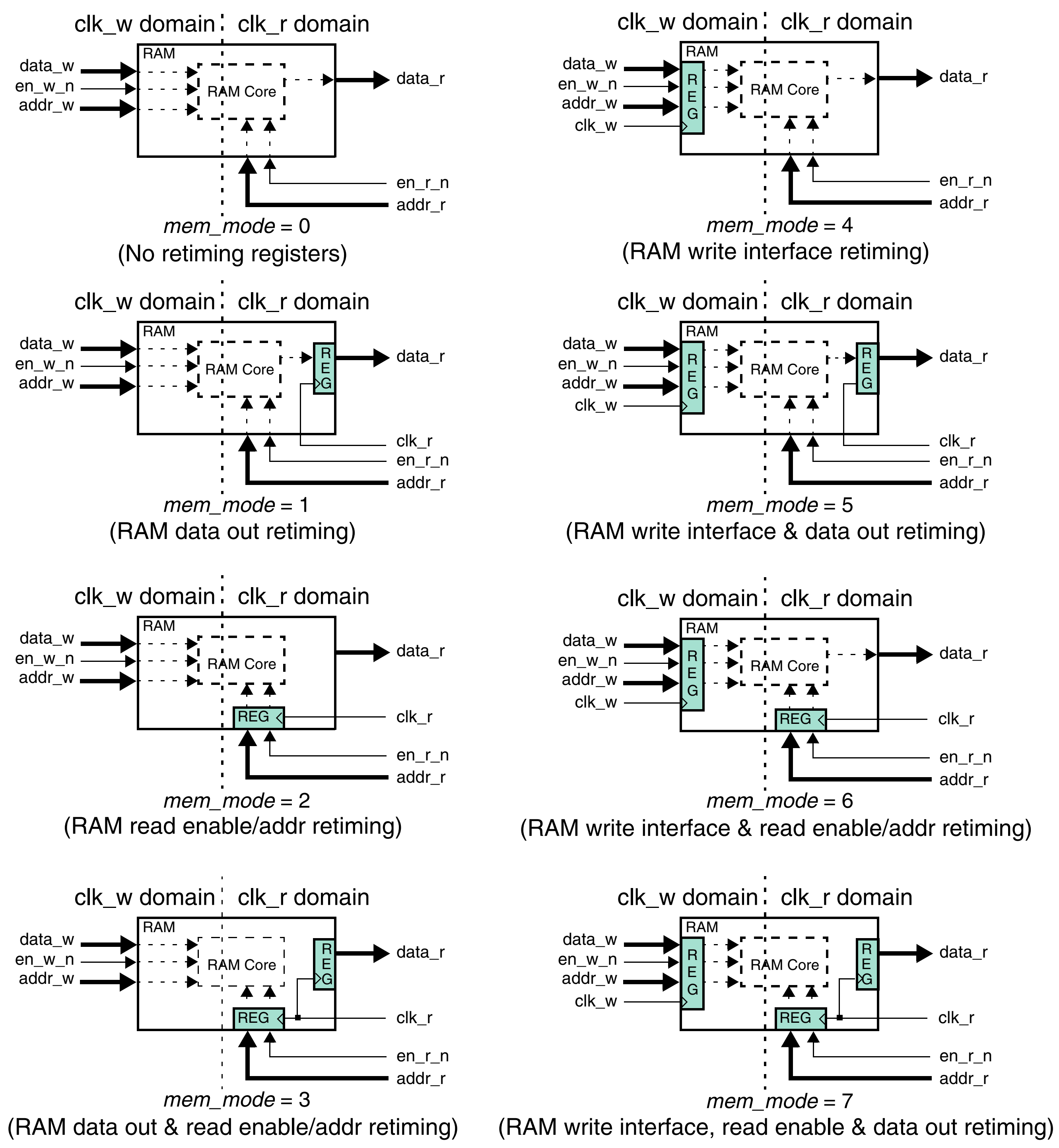
Figure 1-1 Block Diagram of Building Blocks 2-Clock RAM


en_w_n
data_w
width
addr_w
addr_width
clk_w
rst_w_n
init_w_n
Configurable Retiming Stage (*mem_mode* > 3)
RTL FF-based Register Array
RTL-Based Read Data Multiplexer
Configurable Retiming Stage (*mem_mode* = 1, 3, 5, 7)
Configurable Retiming Stage (*mem_mode* = 2, 3, 6, 7)
To all sequential blocks; active low
data_r_a
data_r
width * depth
addr_width
addr_r
en_r_n
clk_r
rst_r_n
init_r_n

Detailed Description of *mem_mode* Setting

To set the *mem_mode* parameter properly, knowledge of the RAM being used with this component is needed. The following diagrams show the eight possible RAM architectures that can interface with the DW_ram_r_w_2c_dff component and the required *mem_mode* setting for each.

Figure 1-2 ***mem_mode*** **Settings Based On RAM Architecture**


clk_w domain
clk_r domain
RAM
RAM Core
REG
data_w
en_w_n
addr_w
clk_w
data_r
clk_r
en_r_n
addr_r
mem_mode = 0
(No retiming registers)
mem_mode = 4
(RAM write interface retiming)
mem_mode = 1
(RAM data out retiming)
mem_mode = 5
(RAM write interface & data out retiming)
mem_mode = 2
(RAM read enable/addr retiming)
mem_mode = 6
(RAM write interface & read enable/addr retiming)
mem_mode = 3
(RAM data out & read enable/addr retiming)
mem_mode = 7
(RAM write interface, read enable & data out retiming)

Write Operation

The `en_w_n` input is an active low signal that enables data to be written into the RAM array. The event in time when data is written into RAM is dependent on the *mem_mode* parameter setting. If *mem_mode* is set to a value of 3 or less, then when `en_w_n` is low, the RAM is written on the rising edge of `clk_w` into the location defined by `addr_w` with value on `data_w`.

If *mem_mode* is set to a value of 4 or more, a single layer of retiming registers are placed on the `data_w`, `addr_w`, and `en_w` input signals. Then when `en_w_n` is low, the RAM is written on the second rising edge of `clk_w` with the `data_w` and `addr_w` coincident with active `en_w_n`.

For diagrams showing how the *mem_mode* setting impacts the write interface, see Figure 1-2 on page 4.

Read Operation

The RAM array is always being read unless the read input interface is configured to be retimed. The event in time when data is read from the RAM is dependent on the *mem_mode* parameter setting. The reading of the RAM is influenced in two ways: (1) possible retiming of read input interface signals and (2) possible retiming of read data output.

Read Input Signals

The read input interface signals consist of `en_r_n` and `addr_r`. They can be retimed by one register of `clk_r` delay when *mem_mode* is set to a value of 2, 3, 6, or 7. In this case, when `en_r_n` is low, the RAM is accessed for reading on the next rising edge of `clk_r` from the location defined by the `addr_r` value coincident with active `en_r_n`.

If *mem_mode* is set to a value of 0 or 4, the `en_r_n` signal is irrelevant and the RAM is immediately accessed for reading based on the `addr_r` value.

For diagrams showing how the *mem_mode* setting impacts the read input interface, see Figure 1-2 on page 4.

Read Data Output

Once the read operation is initiated and the RAM array is accessed, data is provided to the `data_r` output bus. The event in time when `data_r` becomes valid depends on the setting of the *mem_mode* parameter. The `data_r` output buss is pipelined with a single `clk_r` delay when *mem_mode* is set to a value of 1, 3, 5, or 7. The `data_r` output bus is not pipelined when *mem_mode* is 0, 2, 4, or 6.

For diagrams showing how the *mem_mode* setting impacts pipelining of `data_r`, see Figure 1-2 on page 4.

Reset

The DW_ram_r_w_2c_dff component contains asynchronous and synchronous resets in both the write clock domain and read clock domain. The asynchronous and synchronous resets in the write clock domain are `rst_w_n` and `init_w_n`, respectively. When configured to do so (see "rst_mode" on page 6), these signals clear the RAM array when active. These signals also clear write interface retiming registers when *mem_mode* is set to values of 4 or greater.

The asynchronous and synchronous resets in the read clock domain are `rst_r_n` and `init_r_n`, respectively. These signals clear the read interface retiming registers if they are present based on *mem_mode* values.

rst_mode

The *rst_mode* parameter controls whether to clear the RAM array when `rst_w_n` and/or `init_w_n` are active. When *rst_mode* is '0', the RAM array is cleared during write clock domain reset conditions. When *rst_mode* is '1', clearing of the write clock domain excludes clearing of RAM contents.

For sample waveforms that show how *rst_mode* impacts reset behavior, see "Examples of Reset Behavior" on page 11.

Timing Waveforms

Figure 1-3 shows the write and read flow through the RAM when *mem_mode* is set to '0'. Note that the `en_r_n` signal is not toggling and in the 'high' state and the `data_r` output is still getting updated based on the location in RAM defined be `addr_r`.

Figure 1-3 Write and Read with *mem_mode* = 0

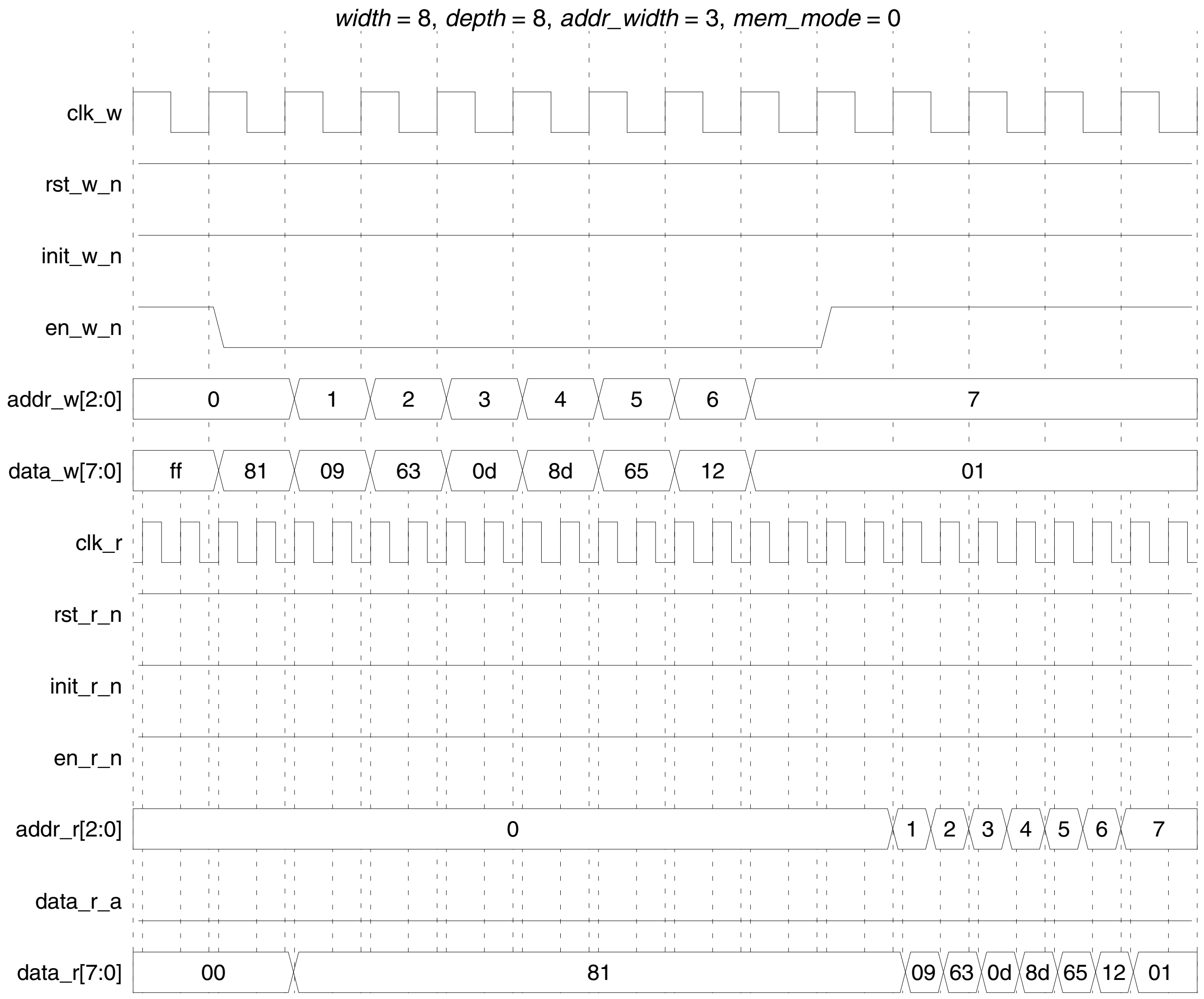


Figure 1-4 shows the write and read flow through the RAM when *mem_mode* = 2. That is, the read input interface signals are retimed before accessing the RAM array. Note that the `en_r_n` signal is in the 'high' state throughout and `data_r` is not being updated as `addr_r` changes. This is because the read interface is being retimed and `en_r_n` needs be sampled as "active" (low) before read access is made. The next timing waveform shows how the activity on `en_r_n` initiates the read operation for *mem_mode* = 2.

Figure 1-4 Write and Attempted Read with *mem_mode* = 2 (No Read Because en_r_n Set "high")

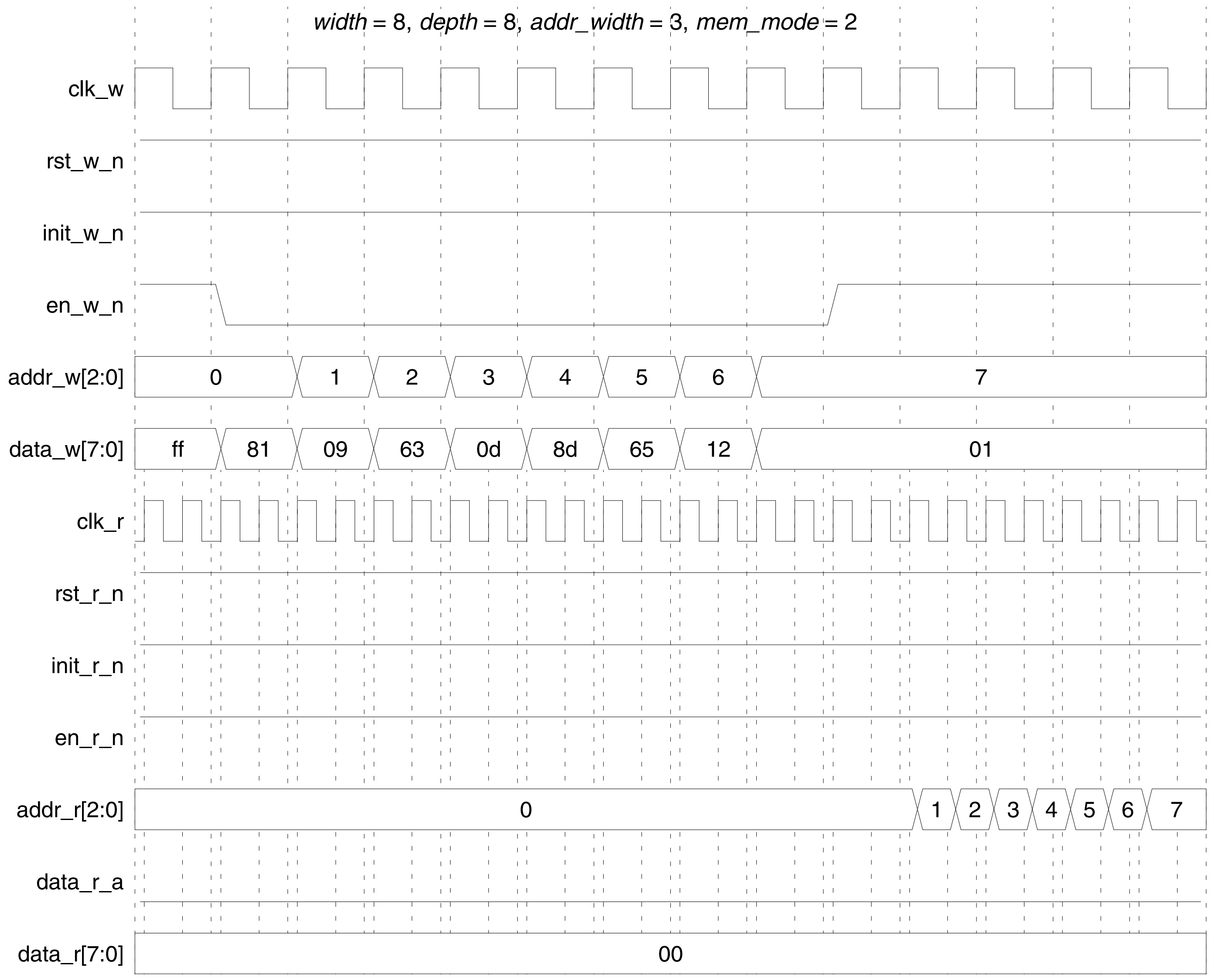


Figure 1-5 shows how `en_r_n` controls the reading of the RAM array for *mem_mode* = 2.

Figure 1-5 Write and Read with *mem_mode* = 2 (en_r_n Must Go "low" To Initiate Read)

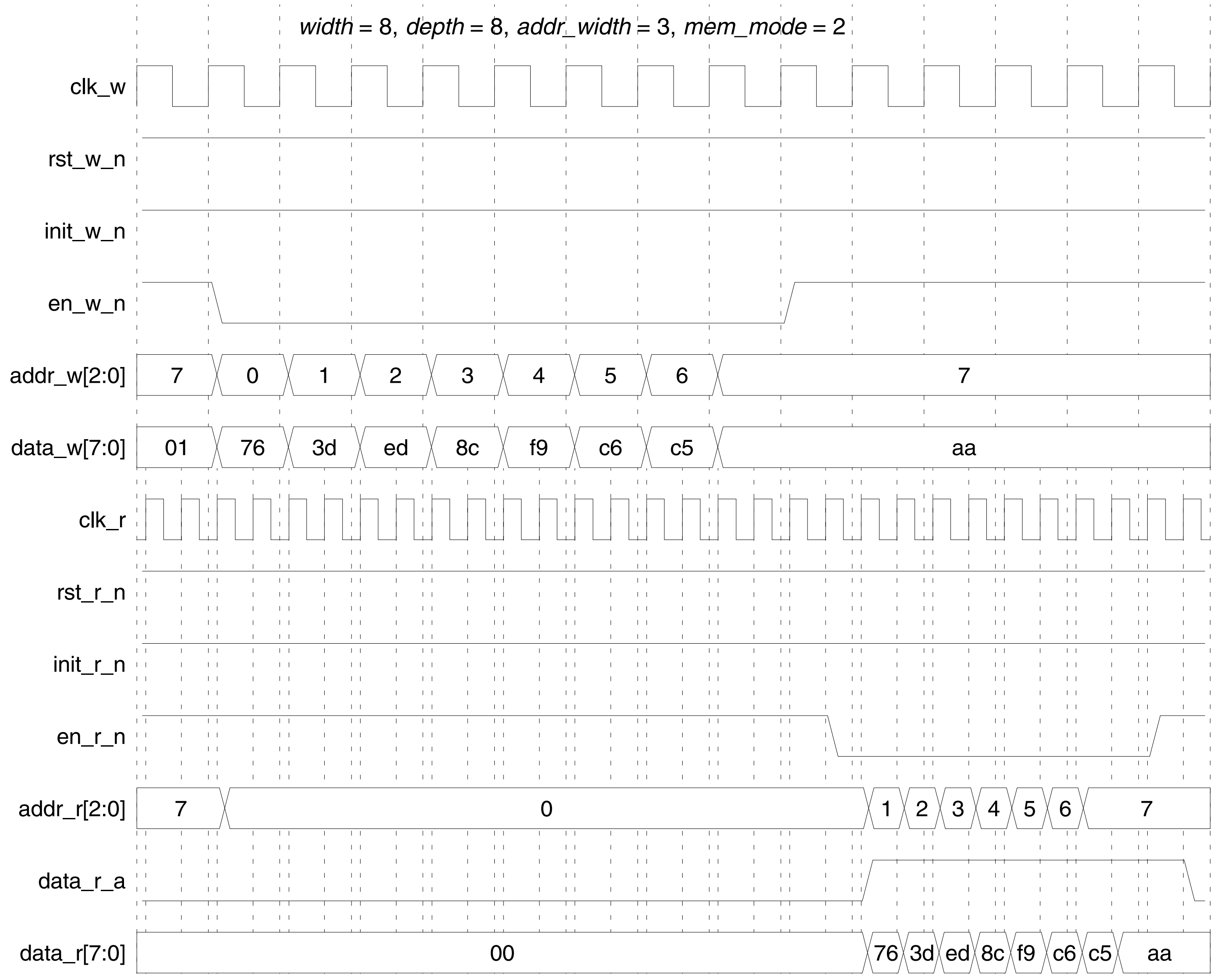


Figure 1-6 shows the behavior of writing and reading the RAM when *mem_mode* = 4. That is, the write interface signals, `en_w_n`, `addr_w`, and `data_w` are retimed before being written into the RAM array which results in a one `clk_w` cycle delay from when *mem_mode* is less than 4. This one `clk_w` cycle delay can be seen as `data_r` occurs later for `addr_r` of '0x0' than timing seen in Figure 1-3 on page 6.

Figure 1-6 Write Operation Behavior When *mem_mode* = 4

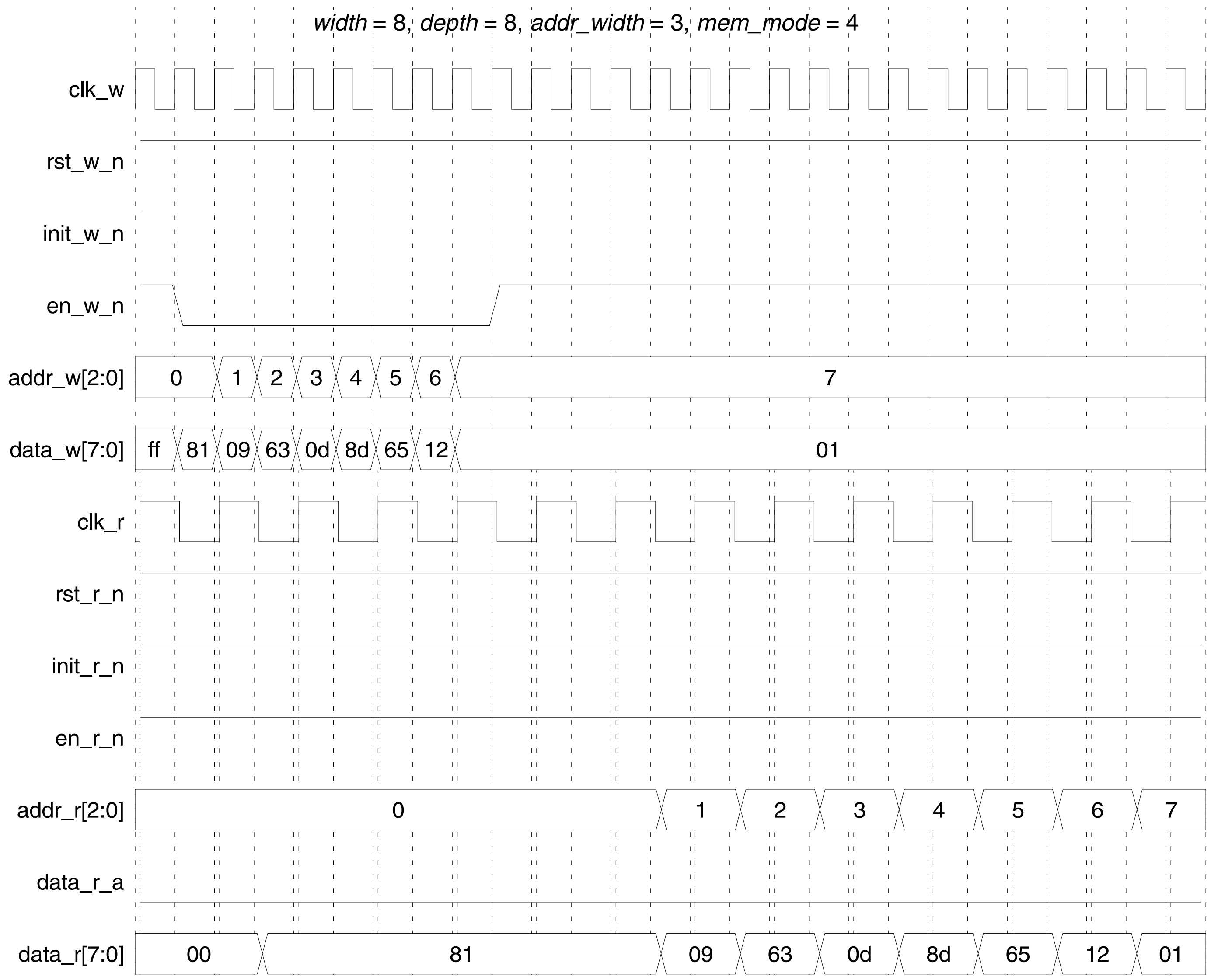
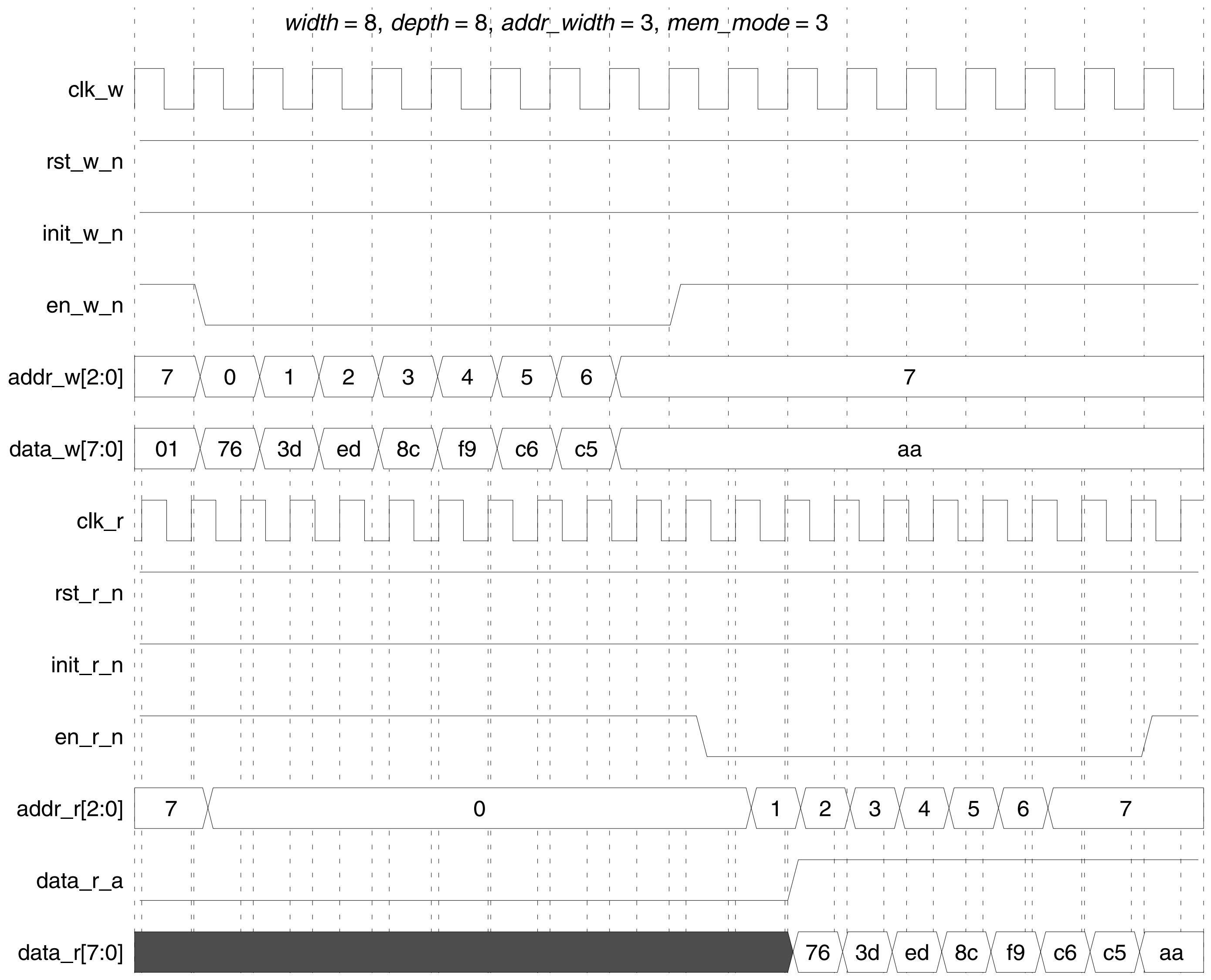


Figure 1-7 shows the read behavior when *mem_mode* = 3.

Figure 1-7 Read Operation Behavior When *mem_mode* = 3



Examples of Reset Behavior

Figure 1-8 shows how the write clock domain asynchronous reset `rst_w_n` clears the RAM contents for *rst_mode* = 0.

Figure 1-8 Asynchronous Reset of RAM for *rst_mode* = 0, *mem_mode* = 4

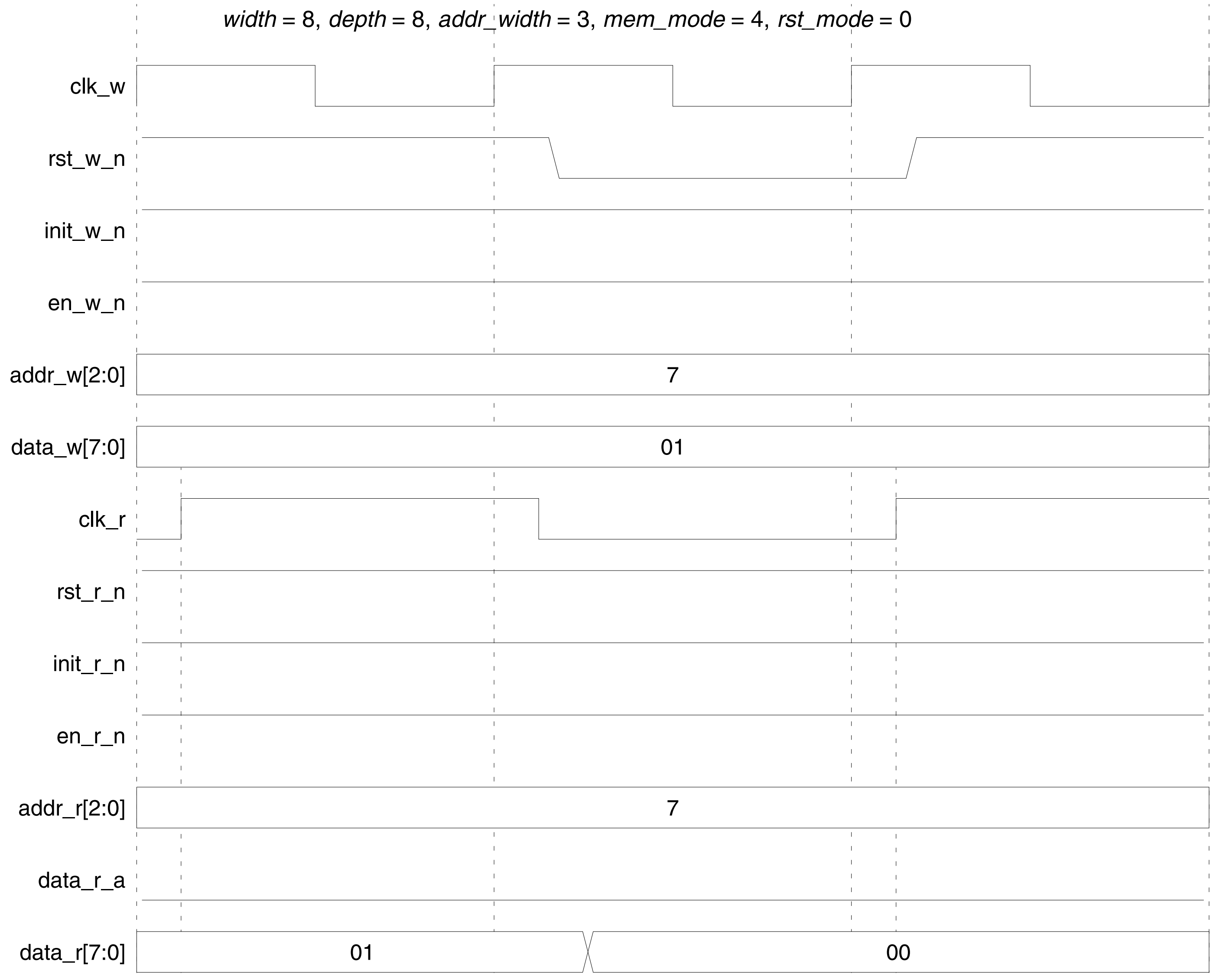
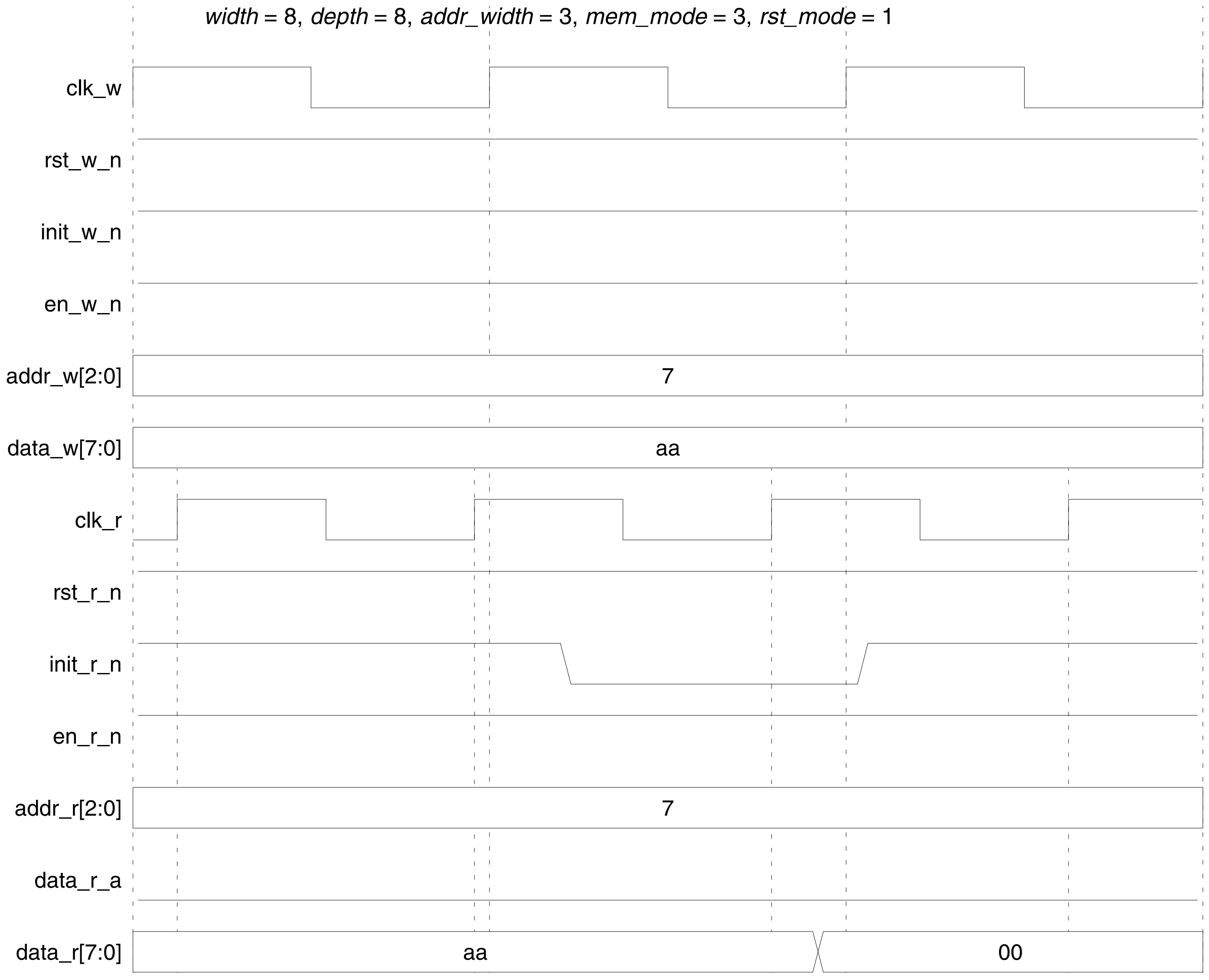


Figure 1-9 shows the affects the read clock domain synchronous reset `init_r_n` has on the read retiming registers when *mem_mode* is 3. In this case *rst_mode* is irrelevant because it only relates to the clearing of the RAM array which is controlled by the write clock domain reset signals. The value of *rst_mode* is specified here as 1 which removes any doubt that the clearing of `data_r` is a function of `init_r_n` going to 0 and not influenced by any other reset signal in either domain.

Figure 1-9 Read Interface Synchronous Reset with *mem_mode* = 3



Related Topics

- Memory – Synchronous RAMs Listing
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,WORK,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW_ram_r_w_2c_dff_inst is
      generic (
        inst_width : POSITIVE := 8;
        inst_depth : POSITIVE := 8;
        inst_addr_width : NATURAL := 3;  -- set to ceil( log2( depth ) )
        inst_mem_mode : NATURAL := 5;
        inst_rst_mode : NATURAL := 1
        );
      port (
        inst_clk_w : in std_logic;
        inst_rst_w_n : in std_logic;
        inst_init_w_n : in std_logic;
        inst_en_w_n : in std_logic;
        inst_addr_w : in std_logic_vector(inst_addr_width-1 downto 0);
        inst_data_w : in std_logic_vector(inst_width-1 downto 0);
        inst_clk_r : in std_logic;
        inst_rst_r_n : in std_logic;
        inst_init_r_n : in std_logic;
        inst_en_r_n : in std_logic;
        inst_addr_r : in std_logic_vector(inst_addr_width-1 downto 0);
        data_r_a_inst : out std_logic;
        data_r_inst : out std_logic_vector(inst_width-1 downto 0)
        );
    end DW_ram_r_w_2c_dff_inst;


architecture inst of DW_ram_r_w_2c_dff_inst is
begin

    -- Instance of DW_ram_r_w_2c_dff
    U1 : DW_ram_r_w_2c_dff
    generic map ( width => inst_width, depth => inst_depth, addr_width =>
inst_addr_width, mem_mode => inst_mem_mode, rst_mode => inst_rst_mode )
    port map ( clk_w => inst_clk_w, rst_w_n => inst_rst_w_n, init_w_n => inst_init_w_n,
en_w_n => inst_en_w_n, addr_w => inst_addr_w, data_w => inst_data_w, clk_r =>
inst_clk_r, rst_r_n => inst_rst_r_n, init_r_n => inst_init_r_n, en_r_n => inst_en_r_n,
addr_r => inst_addr_r, data_r_a => data_r_a_inst, data_r => data_r_inst );

end inst;

-- pragma translate_off
configuration DW_ram_r_w_2s_dff_inst_cfg_inst of DW_ram_r_w_2s_dff_inst is
```

```
    for inst
      end for; -- inst
    end DW_ram_r_w_2s_dff_inst_cfg_inst;
  -- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_ram_r_w_2c_dff_inst( inst_clk_w, inst_rst_w_n, inst_init_w_n, inst_en_w_n,
inst_addr_w,
          inst_data_w, inst_clk_r, inst_rst_r_n, inst_init_r_n, inst_en_r_n,
          inst_addr_r, data_r_a_inst, data_r_inst );

parameter width = 8;
parameter depth = 8;
parameter addr_width = 3;  // set to ceil( log2( depth ) )
parameter mem_mode = 5;
parameter rst_mode = 1;


input inst_clk_w;
input inst_rst_w_n;
input inst_init_w_n;
input inst_en_w_n;
input [(addr_width)-1 : 0] inst_addr_w;
input [width-1 : 0] inst_data_w;
input inst_clk_r;
input inst_rst_r_n;
input inst_init_r_n;
input inst_en_r_n;
input [(addr_width)-1 : 0] inst_addr_r;
output data_r_a_inst;
output [width-1 : 0] data_r_inst;

    // Instance of DW_ram_r_w_2c_dff
    DW_ram_r_w_2c_dff #(width, depth, addr_width, mem_mode, rst_mode)
      U1 ( .clk_w(inst_clk_w), .rst_w_n(inst_rst_w_n), .init_w_n(inst_init_w_n),
.en_w_n(inst_en_w_n), .addr_w(inst_addr_w), .data_w(inst_data_w), .clk_r(inst_clk_r),
.rst_r_n(inst_rst_r_n), .init_r_n(inst_init_r_n), .en_r_n(inst_en_r_n),
.addr_r(inst_addr_r), .data_r_a(data_r_a_inst), .data_r(data_r_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

Date	**Release**	**Updates**
October 2018	DWBB_201806.3	■ First version of this datasheet

Copyright Notice and Proprietary Information